Data Reconstruction: When You See It and When You Don't

Edith Cohen
Google Research, Mountain View, CA, USA
Tel Aviv University, Israel

Haim Kaplan
Tel Aviv University, Israel
Google Research, Tel Aviv, Israel

Yishay Mansour
Tel Aviv University, Israel
Google Research, Tel Aviv, Israel

Shay Moran
Technion, Haifa, Israel
Google Research, Tel Aviv, Israel

Kobbi Nissim
Georgetown University, Washington, DC, USA
Work done while at Google Research,
Tel Aviv, Israel

Uri Stemmer
Tel Aviv University, Israel
Google Research, Tel Aviv, Israel

Eliad Tsfadia
Georgetown University, Washington, DC, USA

Abstract

We revisit the fundamental question of formally defining what constitutes a *reconstruction attack*. While often clear from the context, our exploration reveals that a precise definition is much more nuanced than it appears, to the extent that a single all-encompassing definition may not exist. Thus, we employ a different strategy and aim to "sandwich" the concept of reconstruction attacks by addressing two complementing questions: (i) What conditions guarantee that a given system is protected against such attacks? (ii) Under what circumstances does a given attack clearly indicate that a system is not protected? More specifically,

- We introduce a new definitional paradigm – *Narcissus Resiliency* – to formulate a security definition for protection against reconstruction attacks. This paradigm has a self-referential nature that enables it to circumvent shortcomings of previously studied notions of security. Furthermore, as a side-effect, we demonstrate that Narcissus resiliency captures as special cases multiple well-studied concepts including differential privacy and other security notions of one-way functions and encryption schemes.
- We formulate a link between reconstruction attacks and *Kolmogorov complexity*. This allows us to put forward a criterion for evaluating when such attacks are convincingly successful.

2012 ACM Subject Classification Security and privacy → Human and societal aspects of security and privacy

Keywords and phrases differential privacy, reconstruction

Digital Object Identifier 10.4230/LIPIcs.ITCS.2025.39

Related Version *Full Version*: `https://arxiv.org/abs/2405.15753` [17]

Funding *Edith Cohen*: Partially supported by Israel Science Foundation (grant 1595/19 and 1156/23).
Haim Kaplan: Partially supported by Israel Science Foundation (grant 1595/19 and 1156/23) and the Blavatnik Family Foundation.
Yishay Mansour: Partially funded from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 882396), by the Israel Science Foundation (grant number 993/17), Tel Aviv University Center for AI and Data Science (TAD), and the Yandex Initiative for Machine Learning at Tel Aviv University.
Shay Moran: A Robert J. Shillman Fellow; he acknowledges support by ISF grant 1225/20, by BSF grant 2018385, by an Azrieli Faculty Fellowship, by Israel PBC-VATAT, by the Technion Center for Machine Learning and Intelligent Systems (MLIS), and by the European Union (ERC, GENERALIZATION, 101039692). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

Kobbi Nissim: Partially funded by NSF grant No. 2217678 and by a gift to Georgetown University.
Uri Stemmer: Partially supported by the Israel Science Foundation (grant 1419/24) and the Blavatnik Family foundation.
Eliad Tsfadia: Supported by a gift to Georgetown University.

Acknowledgements The authors would like to thank Noam Mazor for useful discussions about Kolmogorov complexity.

1 Introduction

Reasoning about data privacy is crucial in today's data-driven world. This includes the design of privacy-enhancing technologies aimed at protecting privacy, as well as identifying vulnerabilities of existing methods by conducting "privacy attacks". The most severe family of attacks is, arguably, *reconstruction attacks*, where an adversary takes what appears to be benign statistics and reconstructs significant portions of the sensitive data [19]. Such attacks have helped shape the theory of data privacy as well as expose vulnerabilities in existing real-world systems.

However, prior works did not coalesce around a single *definition of reconstruction* and instead considered several context-dependent definitions. Although these definitions made sense in the context in which they were introduced, they do not necessarily carry over to other settings, as we will later explain. Motivated by this, our work is set to answer the following meta-question:

▶ **Question 1.** *What is reconstruction?*

This question is more nuanced than it initially appears, as will become clear later. It seems that for every attempt to define reconstruction mathematically, there are cases that either fit the definition but do not "feel like" reconstruction, or vice versa. As a result, we do not know whether a precise answer to Question 1 exists, and leave it as an open question for future work. In this work, we aim to approach Question 1 by "sandwiching" the concept of reconstruction and studying the following two questions:

▶ **Question 2.** *What conditions are sufficient to ensure that a given system is protected against reconstruction attacks?*

▶ **Question 3.** *Under what circumstances does an attack clearly indicate that a given system is vulnerable?*

This is reminiscent of the current state of affairs in the related concept of *differential privacy (DP)* [20]: If an algorithm satisfies DP (with small enough privacy parameters), then it is guaranteed to be "safe" in terms of its privacy implications. However, the fact that an algorithm does not satisfy DP does not immediately mean that it is unsafe privacy-wise. Rather, to convince that an algorithm is unsafe, one usually needs to demonstrate an actual attack, such as a reconstruction or a membership attack. In other words, in the context of data privacy, we currently do not have a good definition of what constitutes "privacy"; only definitions of what it means to "protect" privacy, and definitions for what it means to "attack" an algorithm in order to show a privacy breach. We embrace this way of thinking and leverage it towards formally reasoning about reconstruction, as stated in Questions 2 and 3. We elaborate on each of these two questions separately, in Sections 1.1 and 1.2, respectively.

1.1 Protecting against reconstruction

Consider an algorithm $\mathcal{M}$ that takes a dataset S and returns an output y. For example, the dataset S might contain images, and the output y may be an image generative model. This output y is then given to a "privacy attacker" $\mathcal{A}$ whose goal is to output a "reconstruction" z of elements from S. Informally, we want to say that algorithm $\mathcal{M}$ *prevents reconstruction attacks* if y is "hard to invert" in the sense that y does not help the attacker in "recovering" elements from S. However, there are two immediate issues that require attention here:

1. What it means to "recover elements" is context-dependent. For example, our criteria for when one image "recovers" another image would likely differ from our criteria for when a text file "recovers" another text file. As we aim for a general treatment, we abstract this criteria away in the form of a *relation* R, where $R(S, z) = 1$ if and only if z is a "valid reconstruction" of S (or of some data point in S). In our example with the images, this could mean that z is an image which is "close enough" to one of the images in S. Alternatively, depending on the context, this could also mean that z is a collection of 100 images of which one is "close enough" to an image in S, or it could mean that z is a vector of 100 images, each of them is "somewhat close" to a distinct image in S, etc.
2. The underlying distribution of the data matters a lot. For example, consider a scenario where every sufficiently large image dataset contains the Mona Lisa. An "attack" that outputs an image that is similar or identical to the Mona Lisa is not necessarily a successful attack, as this could be accomplished without even accessing the generative model y.

These considerations led [4] and [18] to present definitions with the following flavor:

▶ **Definition 4** ([4, 18]). *Let $\mathcal{X}$ be a data domain, let $\mathcal{D}$ be a distribution over datasets containing elements from $\mathcal{X}$, and let $R : \mathcal{X}^* \times \{0,1\}^* \to \{0,1\}$ be a reconstruction criterion. Algorithm $\mathcal{M}$ is $(\varepsilon, \delta, \mathcal{D})$-$R$-reconstruction-robust if for all attackers $\mathcal{A}$ it holds that*

$$\Pr_{\substack{S\leftarrow\mathcal{D}\\ y\leftarrow\mathcal{M}(S)\\ z\leftarrow\mathcal{A}(y)}}[R(S,z)=1] \leq e^{\varepsilon}\cdot\sup_{z^*}\Pr_{T\leftarrow\mathcal{D}}[R(T,z^*)=1]+\delta. \tag{1}$$

In words, in Definition 4 the attacker's probability of success is compared to a *baseline* which is the probability of success of the best *trivial attacker* that simply chooses a fixed element z^* (independently of the dataset). To contradict the security of algorithm $\mathcal{M}$, the attacker must succeed with a probability noticeably higher than the baseline (depending on the parameters ε and δ). Note that the aforementioned attacker that "recovered" the Mona Lisa would *not* contradict the security of $\mathcal{M}$. The reason is that if every (large enough) dataset sampled from $\mathcal{D}$ would contain the Mona Lisa, then in Definition 4 we could take z^* to be the Mona Lisa, and hence the right hand size of Inequality (1) would be one.[1]

This introduces a serious problem: once the baseline probability is 1 then *no adversary* could ever contradict Inequality (1), even adversaries that do achieve meaningful reconstructions. To illustrate the issue, suppose that the dataset contains both "canonical" photos (such as the Mona Lisa or canonical photos of US presidents), as well as "sensitive" photos of ordinary individuals. While recovering a canonical photo should not be considered a successful attack, recovering a "sensitive" photo definitely should. Formally, suppose that every image in S is sampled independently as follows: with probability 1/2 return the Mona Lisa, and otherwise return a photo of a random citizen. If under these conditions $\mathcal{A}(\mathcal{M}(S))$

[1] For this example we assume that if $z^* \in T$ then $R(T, z^*) = 1$.

is able to recover a photo of an ordinary individual from S, then that should be considered a reconstruction attack. But this is not captured by Definition 4 as the performance of $\mathcal{A}$ is compared with a baseline where the Mona Lisa is recovered. More generally, a sever flaw of Definition 4 is that once the baseline probability is ≈ 1 then no adversary could ever contradict Inequality (1).

1.1.1 Towards a new definition

A takeaway from the above discussion is that comparing all adversaries to the same fixed baseline can create a problem. In particular, adversaries that identify "special" elements in S (which are unlikely to appear in fresh datasets) should be distinguished from adversaries that identify "trivial" elements in S. We now make an attempt to incorporate this into the definition. As we will see, this attempt has different shortcomings.

▶ **Definition 5.** *Let $\mathcal{X}$ be a data domain, let $\mathcal{D}$ be a distribution over datasets containing elements from $\mathcal{X}$, and let $R : \mathcal{X}^* \times \{0,1\}^* \to \{0,1\}$. Algorithm $\mathcal{M}$ is $(\varepsilon, \tau, \mathcal{D})$-R-reconstruction-robust if for all attackers $\mathcal{A}$ it holds that*

$$\Pr_{\substack{S\leftarrow\mathcal{D}\\ y\leftarrow\mathcal{M}(S)\\ z\leftarrow\mathcal{A}(y)}}\Big[R(S,z)=1 \;\; \textit{and} \;\; \Pr_{T\leftarrow\mathcal{D}}[R(T,z)=1]\leq\tau\Big]\leq\varepsilon. \tag{2}$$

In words, with this definition the attacker's goal is to identify an element z such that (1) z is a valid reconstruction w.r.t. the dataset S; and (2) the same z is unlikely to be a valid reconstruction w.r.t. a fresh dataset T. Informally, $\Pr_{T\leftarrow\mathcal{D}}[R(T,z)=1]$ serves as a "conditional baseline" that adapts itself to the element z chosen by the attacker. Note that in the context of our example with the Mona Lisa, an attacker that "recovers" the Mona Lisa would not contradict Inequality (2), because $\Pr_{T\leftarrow\mathcal{D}}[R(T,z)=1]$ would be large and hence greater than the threshold τ. On the other hand, an attacker that manages to recover the photo of an ordinary individual from S (with large enough probability) would contradict Inequality (2). So Definition 5 achieves the desired behavior for this example.

A shortcoming of Definition 5 is that the values of ε and τ are necessarily context dependent. To illustrate this, we will now describe two situations where in one of them we need to set $\varepsilon, \tau < \frac{1}{2}$ in order for the definition to make sense, while in the other situation ε and τ must be larger than $\frac{1}{2}$.

- For the first situation, consider the following distribution $\mathcal{D}$ over datasets: With probability $\frac{1}{2}$ return a dataset containing random photos of ordinary citizens. Otherwise, return a dataset sampled similarly, except that one of its photos is replaced with the Mona Lisa. Under these conditions, we must set $\tau < \frac{1}{2}$ in order to circumvent the "trivial attack" using the Mona Lisa. Setting ε close to 0 seems reasonable under these conditions in order to guarantee that a meaningful reconstruction can happen only with small probability.
- For the second situation, consider a case where the dataset is a vector of n random bits, and the attacker's goal is to pinpoint a single entry from this vector and to guess it with high probability. Formally, in this example we interpret the outcome of the adversary as $z = (i, b) \in [n] \times \{0,1\}$, where $R(S,(i,b)) = 1$ if and only if $S[i] = b$. In this scenario, *every* adversary always satisfies $\Pr_{T\leftarrow\mathcal{D}}[R(T,z)=1] = 1/2$ and hence if we set $\tau < 1/2$ then every algorithm is safe w.r.t. to that definition, which is absurd. Therefore we may assume that $\tau \geq 1/2$. In this case the condition $\Pr_{T\leftarrow\mathcal{D}}[R(T,z)=1] \leq \tau$ holds for every adversary and hence is redundant and we are only left with the first condition. How about ε? Note that a trivial adversary, which always guesses that the first bit is 1,

succeeds with probability $1/2$. Hence, ε must be $\geq 1/2$ or else no algorithm can be safe. To summarize, if $\tau < 1/2$ then every algorithm is safe, and else, if $\tau \geq 1/2$ and $\varepsilon < 1/2$ then no algorithm is safe. Hence, this condition is non-trivial only in the range $\varepsilon, \tau \geq 1/2$.

Intuitively, the issue about ε, τ being context dependent stems from their somewhat nonstandard semantics (especially the parameter τ). Ideally, we would want ε and τ to quantify some notion of "distance from optimality" (similarly to common definitions in the literature, such as differential privacy). But this is not the case with Definition 5. Rather, the threshold τ dictates what we consider to be a "non-trivial" baseline. But "non-trivial" is context dependent, and hence so is τ.

1.1.2 A new Definitional paradigm: Narcissus resiliency – an adversary trying to beat itself in its own game

So far, we have established that: (1) We need to compare the adversary's success probability to a baseline; (2) This baseline cannot be fixed for all adversaries; and (3) We want to circumvent the need for a hyperparameter controlling what "non-trivial" is, as this is likely to be context dependent. To tackle this, we present a new definitional paradigm which we call *Narcissus resiliency*. In this paradigm, instead of explicitly setting and tuning the baseline, we *let the attacker be its own baseline*.[2] More specifically, the attacker's success probability on the real dataset is compared with its success probability in a baseline setting where it does not get any information about the real dataset. Furthermore, the attacker must use the same strategy in both settings. This is enforced by preventing the attacker from receiving any information that can help it distinguish whether it is examined with respect to real dataset S or a fresh dataset T sampled from the same distribution. Hence, a Narcissus attacker needs to choose a compromise strategy: on one hand it needs to be strong (so it succeeds with high probability when it is applied to S) and at the same time it needs to be weak (so it succeeds with low probability when it is applied to T). The formal definition follows:[3]

▶ **Definition 6** (Narcissus resiliency). *Let $\mathcal{X}$ be a data domain, let $\mathcal{F}$ be a family of distributions over datasets containing elements from $\mathcal{X}$, and let $R : \mathcal{X}^* \times \{0,1\}^* \to \{0,1\}$. Algorithm $\mathcal{M}$ is $(\varepsilon, \delta, \mathcal{F})$-R-Narcissus-resilient if for all $\mathcal{D} \in \mathcal{F}$ and for all attackers $\mathcal{A}$ it holds that*

$$\Pr_{\substack{S\leftarrow\mathcal{D}\\ y\leftarrow\mathcal{M}(S)\\ z\leftarrow\mathcal{A}(y)}}[R(S,z)=1] \leq e^{\varepsilon} \cdot \Pr_{\substack{S\leftarrow\mathcal{D}\\ T\leftarrow\mathcal{D}\\ y\leftarrow\mathcal{M}(S)\\ z\leftarrow\mathcal{A}(y)}}[R(T,z)=1] + \delta. \tag{3}$$

Note that in both experiments the adversary $\mathcal{A}$ is executed on a dataset S sampled exactly in the same way. Thus, the adversary behaves exactly the same way in both executions. Nevertheless, the adversary's goal is to "separate" the two experiments, where in the left experiment it aims to maximize the probability of reconstruction while in the right experiment it aims to minimize this probability.

▶ **Example 7.** Note that Definition 6 achieves the desired behavior in our example with the Mona Lisa: In order to contradict Inequality (3), the attacker must recover (with noticeable probability) the photo of a citizen from S.

[2] The name "Narcissus" is inspired by the Greek myth of Narcissus, who became obsessed with his own reflection. This alludes to how, in our proposed paradigm, the adversary competes against itself rather than external benchmarks.

[3] In the following definition we switch from a single data distribution $\mathcal{D}$ to a family of data distributions $\mathcal{F}$, where the mechanism is required to be "secure" w.r.t. every distribution in the family. This is standard, and can also be applied in the context of Definition 4.

▶ **Example 8.** Let the underlying dataset distribution $\mathcal{D}$ be uniform over n-bit vectors. Let $k \in [n]$ be a parameter and suppose that, for the sake of this example, the relation R is defined as follows. Given a dataset $S \in \{0,1\}^n$ and the outcome of the attacker z, parse z as a vector of k distinct indices $\tilde{I} = (i_1, \ldots, i_k) \in [n]^k$ and k values $\tilde{Z} = (\tilde{z}_1, \ldots, \tilde{z}_k) \in \{0,1\}^k$ and let $R(S, z) = 1$ if and only if $S|_{\tilde{I}} = \tilde{Z}$. That is, in this example, the adversary's goal is to pinpoint k coordinates from S and to guess them correctly. In order to contradict Inequality (3), the attacker must recover the values of k coordinates from S with probability noticeably higher than 2^{-k}.

We arrived at Definition 6 through the lens of data reconstruction. To provide further evidence supporting the paradigm of Narcissus resiliency, we show in Section 2 that Narcissus resiliency captures concepts from cryptography and privacy as special cases. This includes differential privacy, resiliency to membership inference attacks, security of one-way functions, security of encryption schemes, and more. We show that all of these concepts can be stated in the terminology of Narcissus resiliency, where an adversary is "trying to beat itself in its own game". Furthermore, the paradigm enables us to express strictly weaker protections compared to differential privacy that are still meaningful. To support this claim, in the full version [17] we provide an example of a very natural *deterministic* mechanism (which is clearly not differentially private) that estimates a counting query under Narcissus-resiliency w.r.t. the family of all i.i.d. distributions and a natural reconstruction predicate R.

1.2 Identifying reconstruction

So far we have discussed which types of mechanisms prevent reconstruction of the input data. Such definitions serve as important guidance for responsible algorithm design. However, to show that a mechanism is vulnerable to reconstruction in the real world, an attacker usually only has access to the output of the mechanism, which in the learning context is a model that can be seen as an interactive program, or even just as a fixed string (e.g., a real-world attacker would like to attack a given chat-bot model, and not the algorithm that created the model based on training data). Moreover, such attacker might not even be aware of the training procedure exactly.

Mathematically, in Section 1.1 we considered a setting where initially a dataset S is sampled, and then an outcome y is computed by an algorithm $\mathcal{M}$ based on S. Our definition of Narcissus resiliency 6 pinpoints our ultimate desire from the algorithm $\mathcal{M}$. Often, however, $\mathcal{M}$ is not explicit and we only have access to a fixed model y and a fixed training set S. So there is no clear distribution over S and it is not clear what is the process which was used to compute y from S. Still, in order to be able to assess whether a particular attack is successful, we would like to have a definition of what it means to "reconstruct" a particular S from a particular y. To emphasize this difference, when S and y are fixed and we want to classify a successful attack (rather than resiliency to such) we use the term "extract" instead of "reconstruct".

▶ **Question 9.** *How do we define that a* fixed *dataset S (or a* fixed *point in it $x \in S$) is extractable from a* fixed *string y?*

Intuitively, such "extraction" means that x is encoded in y. But capturing this intuition more formally turns out to be challenging. In this work, we offer the first formal definition that, we hope, will help understand what real-world attacks do, and what properties future attacks should highlight in order to argue about their quality.

Prior works presented several definitions with the following flavor:

▶ **Definition 10** ([14, 11]). *Let R be a relation such that $R(x,z)=1$ means that z is a valid extraction of x. A string x is* extractable *from a model y, if there exists an efficient program $\mathcal{A}$ such that $R(x,\mathcal{A}(y))=1$.*

Intuitively, the program $\mathcal{A}$ in Definition 10 serves as evidence that x is extractable from y, because given y it outputs z with $R(x,z)=1$. For example, if x appears at the i^{th} location of y, then the program that given y as input, outputs $y_i, y_{i+1}, \ldots, y_{i+|x|-1}$ satisfy the condition of Definition 10.

However, at the formal level, this definition is meaningless: Because $\mathcal{A}$ is chosen *after* the example x, then formally, every example x is "extractable" as there exists $\mathcal{A}$ that (ignores its input) and outputs x. [14] mentioned that in order to prevent such pathological cases, the program $\mathcal{A}$ should be *shorter* than x (and therefore $\mathcal{A}$ cannot "memorize" x). But this by itself is not enough. For example, consider the string $x = \underbrace{34\,34\,\ldots\,34}_{1000\text{ times}}$ and the program $\mathcal{A}$ "Print '34' 1000 times". Namely, this example illustrates that $\mathcal{A}$ being short is clearly not evidence for not being able to "memorize" x. Therefore, the length of $\mathcal{A}$ cannot be the only criteria for determining its validity, and there should be some connection between the length of the attack $\mathcal{A}$, and the "complexity" of the target output. Intuitively, if there is no way to encode the target output x using a short program (i.e., to compress it), then by demonstrating a short attacking program that is able to reveal x given y, proves that x is really extracted from the model y.

▶ **Question 11.** *How do we quantify the "complexity" of an outcome x?*

This question leads us to a classical complexity measure, called *Kolmogorov-Complexity*.

1.2.1 Defining Extraction via Kolmogorov Complexity

What makes the string 343434343434343 less random than 285628563123452? The notion of *Kolmogorov complexity* (in short, K-complexity), introduced by [56, 37, 15] in the 60s, provides an elegant method for measuring the amount of "randomness" in an individual string. Informally, the K-complexity of a string x, denoted by $K(x)$, is the length of the shortest program that outputs the string x. Intuitively, a string x has high K-complexity, if there is no short program that outputs it and halts.

One issue with K-complexity is that it is not well defined without specifying which programming language $\mathcal{L}$ do we use. In theoretical results (when constants do not matter), this complexity is usually defined w.r.t. a fixed *Universal Turing Machine*. But when constants matter, a more formal way to define it is as follows.

▶ **Definition 12** ($K_{\mathcal{L}}$-Complexity). *Let $\mathcal{L}$ be a programming language (e.g., Python). The $K_{\mathcal{L}}$-complexity of a string x, denoted by $K_{\mathcal{L}}(x)$, is the length of the shortest $\mathcal{L}$-program that outputs x and halts. Similarly, given a set of strings X, we denote by $K_{\mathcal{L}}(X)$ the length of the shortest $\mathcal{L}$-program that outputs an element in X and halts.*

Now that we have Definition 12 in hand, we are finally ready to define extraction.

▶ **Definition 13** (Our extraction definition, informal). *Let R be an extraction relation and $\mathcal{L}$ a programming language. We say that a string x is $(R,\mathcal{L})$-extractable from a string y iff there exists an $\mathcal{L}$-program $\mathcal{A}$ such that the following holds:*

1. *$\mathcal{A}(y)$ outputs z such that $R(x,z)=1$, and*
2. *$K_{\mathcal{L}}(\{z\colon R(x,z)=1\}) \gg |\mathcal{A}|$.*

We measure the quality *of the extraction by $1-\frac{|\mathcal{A}|}{K_{\mathcal{L}}(\{z\colon R(x,z)=1\})}$ (closer to 1 means a more significant extraction).*

Namely, x is "$(R, \mathcal{L})$-extractable" from y if there exists a *short* $\mathcal{L}$-program $\mathcal{A}$ that outputs an extraction (according to R) of x from y, while there is no short $\mathcal{L}$-program that extracts x without y. We note that extraction must be in a context of a specific programming language, since otherwise, for every x and y we could always find a "special" programming language $\mathcal{L}_{x,y}$ such that x is $\mathcal{L}_{x,y}$-extractable from y.[4] Satisfying this definition w.r.t. a *standard* programming language (e.g., Python, Java, etc) should indeed be considered as a valid extraction.

Definition 13 can easily be extended to datasets S (rather than a single example x), and in Section A we also consider a probabilistic version of the $K_{\mathcal{L}}$-complexity that allows some error probability, and redefine Definition 13 w.r.t. this version.

To provide further evidence supporting Definition 13, we demonstrate in Section A.3 how three different types of real-world attacks [14, 30, 11] can be explained using the terminology of Definition 13.

1.2.2 Narcissus-Resiliency prevents non-trivial extraction of training data

In Section 3 we show that if $\mathcal{M}$ is $(\varepsilon, \delta, \mathcal{F})$-$R$-Narcissus-resilient then it prevents extraction in the following sense. Fix a distribution $\mathcal{D} \in \mathcal{F}$, sample two independent datasets S and T from $\mathcal{D}$, and compute $y \leftarrow \mathcal{M}(S)$. We treat S as the "real" dataset and T as a "shadow" dataset. Suppose that there is no adversary $\mathcal{B}$ that given y can find a short program $\mathcal{A}$ such that $R(T, \mathcal{A}(y)) = 1$. Informally, this means that extracting information about the shadow dataset T without receiving any information about it is hard. Then, if $\mathcal{M}$ is Narcissus-resilient, there is also no adversary that achieves this w.r.t. S (even though the adversary gets y which was computed based on S).

1.2.3 Verifying the validity of reconstruction attacks

The main limitation of Definition 13 is that Item 2 (lower-bounding the Kolmogorov Complexity) is not verifiable, as computing the K-complexity is an intractable problem. We could consider a tractable version of it, called *time-bounded* K-complexity, where given a parameter t we consider only programs that halt within t steps ([37, 55, 57, 36]). This, however, would only relax intractability to inefficiency, which does not help in our context where we would like to verify a reconstruction attack efficiently. In practice, we can only use Heuristics to gain some level of confidence. E.g., to apply many well-known compression algorithms on x and check that all of them results with a compressed representation that is much longer than $|\mathcal{A}|$. Yet, we remark that any such Heuristic can fail to determine some highly compressible patterns, and as we demonstrate in Section A.4.1, this problem is indeed inherent assuming that a basic cryptography primitive (pseudo-random generator) exists.

1.3 Additional related works

Memorization

Perhaps the most basic privacy violation is *memorization*. [25, 24, 6, 7, 43, 1] theoretically study the necessity of memorization in learning. Roughly speaking, they show that for some tasks, the output of any accurate algorithm must have large *mutual information* with the training data. A similar result of [9] show that there exist learning tasks in which

[4] For example, consider a programming language $\mathcal{L}_{x,y}$ that given a command y outputs x.

memorization is only necessary for *efficient* learners. While large memorization (i.e., mutual information) implies that the algorithm is not differentially private, it does not imply the existence of an efficient attack. Indeed, except for [9, 1], these works are typically not constructive (i.e., they do not provide an efficient way to translate the memorization into an efficient privacy attack).

Computational Differential Privacy

[5, 48] considered a computational relaxation of (the standard, information-theoretic) differential privacy, where they require that the outputs of two executions on neighboring datasets are indistinguishable only from the eyes of an *efficient* observer. This relaxation turns out to be necessary for fundamental distributed tasks [47, 31] but also for some (artificial) centralized ones [8, 27]. We note that an efficient privacy attack (e.g., membership inference or reconstruction) implies that the algorithm is not computationally differentially private. But in the opposite direction, if an algorithm is not computationally differentially private, the privacy attack, while being efficient, could be very negligible (e.g., the attack might reveal only a single, insignificant, bit of information about a training example, and still violating computational differential privacy).

Membership Inference

In *membership inference (MI)* [53, 60], an adversary is given a target example and its goal is to infer whether it was included in a model's training set. Most techniques follow a paradigm of measuring some correlation/loss function between the target example and the model and checking if it exceeds some threshold. For example, [22] showed that any algorithm that given $x_1, \ldots, x_n \in \{-1, 1\}^d$ outputs $y \in [-1, 1]^d$ that is sufficiently close to the average $\frac{1}{n}\sum_{i=1}^{n} x_i$, is exposed to the MI attack that given a target example x, decides "IN" or "OUT" based on whether $\langle x, y\rangle$ (the inner-product between x and y) exceeds some threshold. The MI literature is very rich, and includes works that perform attacks on fundamental learning tasks (e.g., [52, 22, 50, 2, 1]) along with MI attacks on more complex models (e.g., [54, 60, 34, 10, 44, 50, 58]). We remark that while MI attacks are indeed a reasonable privacy concern in many scenarios, they are weaker than reconstruction attacks because the attacker needs to know the target training example beforehand while without it there might be no way to extract sensitive information.

Reconstruction

The results of [19] provided a foundation for rigorously quantifying reconstruction bounds for general query release mechanisms. In their setting, the dataset is binary, and they show that given a few statistical queries, it is possible to reveal 90% of the dataset. [21, 23, 31] provide improved results in similar settings like [19] but under weaker accuracy assumptions. In more general settings, [29] demonstrated that both Rényi Differential Privacy (DP) and Fisher information offer robust semantic assurances against reconstruction attacks. [4] introduced the concept of reconstruction robustness (ReRo), establishing a link between reconstruction attacks and DP. [32] extended [4]'s to analyze reconstruction attacks against DP-SGD. [35] furthered this research by examining the connection between the hypothesis testing interpretation of DP (specifically f-DP) and reconstruction robustness. From a more practical standpoint, there is a rich literature on reconstruction attacks on various real-world models (e.g., [26, 12, 59, 33, 13, 61, 11, 30]). In Section A (recognizing reconstruction), we focus on three of them in order to illustrate the expressiveness of our Definition 13.

Formalizing legal concepts of privacy

Some of the work towards formalizing legal privacy concepts mathematically has taken an approach similar to our "sandwiching" of the concept of reconstructing rather than attempting a precise answer to Question 1. These works did not attempt to model a legal concept exactly but rather define requirements which are clearly stricter or clearly weaker than the those of the legal concept, yet non-trivial so they can substantiate a claim that the use of a certain technology satisfies or does not satisfy the legal requirement. As one example, in their modeling of the FERPA privacy requirement,[5] [49] provided a definition which is stronger than the legal requirement, yet satisfiable by the use of differential privacy, hence providing strong evidence that the use of differential privacy (with appropriate parameters) satisfies the legal requirements. As another example is the modeling of the GDPR requirement of protection against singling out [16] defined *protection against predicate singling out*, a concept which is weaker than the legal requirement. Showing that k-anonymity does not protect against predicate singling out they hence claimed that it does neither satisfy the legal requirement.

2 Expressiveness of Narcissus resiliency

In the introduction we presented the definition of Narcissus resiliency (Definition 6) with the interpretation of "security against reconstruction attacks", where the function R encapsulates what a "valid reconstruction" means. We now show that with different instantiations of the function R, the same framework can be used to capture many other existing security notions. In this section we show this for the notions of *membership inference attacks* and for *predicate singling out*. In the full version of this paper [17], we extend this to additional security notions, including *differential privacy*, security of *one-way functions*, and security of *encryption schemes*. The fact that Narcissus resiliency is expressive enough to capture all these (seemingly unrelated) well-established notions enhances our confidence in its application as a security notion against reconstruction attacks.

2.1 Membership inference as Narcissus resiliency

Membership Inference (MI) attacks [53, 60] are a family of attacks which are applied mostly to machine learning models. The goal of an MI attack is to determine whether a given data record was part of the training data underlying the model or not. More specifically, let $\mathcal{D}$ be a distribution over data records, and let $\mathcal{M}$ be an algorithm for analyzing datasets of size n. Algorithm $\mathcal{M}$ is said to be MI-secure if no adversary $\mathcal{A}$ has a significant advantage in distinguishing between the outputs of the following two experiments:

- Sample $S \leftarrow \mathcal{D}^n$ and $z \leftarrow \mathcal{D}$ independently. Let $y \leftarrow \mathcal{M}(S)$. Output (y, z).
- Sample $S \leftarrow \mathcal{D}^n$. Let z be a uniformly random element from S. Let $y \leftarrow \mathcal{M}(S)$. Output (y, z).

The formal definition is as follows.

▶ **Definition 14** (Resilience to Membership Inference, [53, 60]). *Let $\mathcal{M} : \mathcal{X}^n \to Y$ be an algorithm that operates on a dataset, and let $\mathcal{D}$ be a distribution over $\mathcal{X}$. We say that $\mathcal{M}$ is $(\delta, \mathcal{D})$-MI-secure if for every adversary $\mathcal{A}$ it holds that*

[5] FERPA – The Family Educational Rights and Privacy Act (FERPA) – is is a Federal law that protects the privacy in education records.

$$\left| \Pr_{\substack{S\leftarrow\mathcal{D}^n \\ y\leftarrow\mathcal{M}(S) \\ z\leftarrow\mathcal{D} \\ b\leftarrow\mathcal{A}(y,z)}}[b=1] - \Pr_{\substack{S\leftarrow\mathcal{D}^n \\ y\leftarrow\mathcal{M}(S) \\ z\in_{\mathrm{R}}S \\ b\leftarrow\mathcal{A}(y,z)}}[b=1] \right| \leq \delta.$$

We show that Narcissus resiliency captures the concept of membership inference. We make use of the following function:

▶ **Definition 15.** *We define R_{MI} to be a (randomized) binary function, which takes two arguments: a dataset $Z \in \mathcal{X}^n$ and a (possibly randomized) function $f : \mathcal{X} \to \{0,1\}$. Given Z, f, to compute $R_{\mathrm{MI}}(Z,f)$, sample a point $z \in Z$ and return $f(z)$.*

▶ **Theorem 16.** *Let $\mathcal{M} : \mathcal{X}^n \to Y$ be an algorithm and let $\mathcal{D}$ be a distribution over $\mathcal{X}$. Then $\mathcal{M}$ is $(\delta,\mathcal{D})$-MI-secure if and only if it is $(0,\delta,\{\mathcal{D}^n\})$-$R_{\mathrm{MI}}$-Narcissus-resilient.*

Proof. First observe that an MI-adversary gets two arguments (an outcome y and a point z) and returns a bit, while an R_{MI}-Narcissus-adversary gets only one argument (the outcome y) and returns a binary function f that takes a point and returns a bit (aiming to satisfy R_{MI}). To help bridge between these notations, given an MI-adversary $\mathcal{A}$ and an outcome y, we write $f(\cdot) \leftarrow \mathcal{A}(y,\cdot)$ to denote the binary function obtained by fixing $\mathcal{A}$'s first argument to be y. We can interpret this as an R_{MI}-Narcissus-adversary that takes one argument (the outcome y) and returns the function $f(\cdot) \leftarrow \mathcal{A}(y,\cdot)$. It follows that

$$\Pr_{\substack{S\leftarrow\mathcal{D}^n \\ y\leftarrow\mathcal{M}(S) \\ f(\cdot)\leftarrow\mathcal{A}(y,\cdot)}}[R_{\mathrm{MI}}(S,f)=1] - \Pr_{\substack{S\leftarrow\mathcal{D}^n \\ T\leftarrow\mathcal{D}^n \\ y\leftarrow\mathcal{M}(S) \\ f(\cdot)\leftarrow\mathcal{A}(y,\cdot)}}[R_{\mathrm{MI}}(T,f)=1] = \Pr_{\substack{S\leftarrow\mathcal{D}^n \\ y\leftarrow\mathcal{M}(S) \\ z\in_{\mathrm{R}}S \\ b\leftarrow\mathcal{A}(y,z)}}[b=1] - \Pr_{\substack{S\leftarrow\mathcal{D}^n \\ y\leftarrow\mathcal{M}(S) \\ z\leftarrow\mathcal{D} \\ b\leftarrow\mathcal{A}(y,z)}}[b=1].$$

Hence, if $\mathcal{M}$ is $(\delta,\mathcal{D})$-MI-secure then it is also $(0,\delta,\{\mathcal{D}^n\})$-$R_{\mathrm{MI}}$-Narcissus-resilient. In addition, if the above two differences are positive, then the equality holds also in absolute value, and hence the condition of $(0,\delta,\{\mathcal{D}^n\})$-$R_{\mathrm{MI}}$-Narcissus-resiliency implies the condition of $(\delta,\mathcal{D})$-MI-security. Otherwise, to show that the left hand side cannot be smaller than $-\delta$, let $\hat{\mathcal{A}}$ be the R_{MI}-Narcissus-adversary that runs $\mathcal{A}$ and returns the "inverted" function $\hat{f} \equiv 1 - f$. Then,

$$\begin{aligned}
&\Pr_{\substack{S\leftarrow\mathcal{D}^n \\ T\leftarrow\mathcal{D}^n \\ y\leftarrow\mathcal{M}(S) \\ f(\cdot)\leftarrow\mathcal{A}(y,\cdot)}}[R_{\mathrm{MI}}(T,f)=1] - \Pr_{\substack{S\leftarrow\mathcal{D}^n \\ y\leftarrow\mathcal{M}(S) \\ f(\cdot)\leftarrow\mathcal{A}(y,\cdot)}}[R_{\mathrm{MI}}(S,f)=1] \\
&= -\left(\Pr_{\substack{S\leftarrow\mathcal{D}^n \\ T\leftarrow\mathcal{D}^n \\ y\leftarrow\mathcal{M}(S) \\ \hat{f}(\cdot)\leftarrow\hat{\mathcal{A}}(y,\cdot)}}[R_{\mathrm{MI}}(T,\hat{f})=1] - \Pr_{\substack{S\leftarrow\mathcal{D}^n \\ y\leftarrow\mathcal{M}(S) \\ \hat{f}(\cdot)\leftarrow\hat{\mathcal{A}}(y,\cdot)}}[R_{\mathrm{MI}}(S,\hat{f})=1] \right) \\
&= \Pr_{\substack{S\leftarrow\mathcal{D}^n \\ y\leftarrow\mathcal{M}(S) \\ \hat{f}(\cdot)\leftarrow\hat{\mathcal{A}}(y,\cdot)}}[R_{\mathrm{MI}}(S,\hat{f})=1] - \Pr_{\substack{S\leftarrow\mathcal{D}^n \\ T\leftarrow\mathcal{D}^n \\ y\leftarrow\mathcal{M}(S) \\ \hat{f}(\cdot)\leftarrow\hat{\mathcal{A}}(y,\cdot)}}[R_{\mathrm{MI}}(T,\hat{f})=1] \leq \delta.
\end{aligned}$$

which follows from the resilience of $\mathcal{M}$ against $\hat{\mathcal{A}}$. ◀

2.2 Predicate singling out as Narcissus resiliency

[16] defined a type of privacy attack called *Predicate Singling Out (PSO)*, intended to mathematically formulate the legal concept of "singling out" that appears in the General Data Protection Regulation (GDPR). Concretely, let $\mathcal{D}$ be a data distribution and let $S \leftarrow \mathcal{D}^n$ be a dataset containing n iid samples from $\mathcal{D}$. Let $\mathcal{M}$ be an algorithm that operates on S and return an outcome y. A PSO adversary gets y and aims to find a predicate p that "isolates" one record in S, meaning that it evaluates to 1 on *exactly* one record in S. Note, however, that without further restrictions this is not necessarily a hard task. In particular, even without looking at y, the adversary might choose a predicate p whose expectation over $\mathcal{D}$ is $1/n$. Such a predicate would isolate a record in S with constant probability. More generally, if the expectation of p is w, then the probability that it isolates a record in a fresh dataset of size n is $n \cdot w \cdot (1-w)^{n-1}$. Thus, for the attack to be considered "significant", the adversary has to succeed in its attack with probability noticeably higher than this. The formal definition is as follows.

▶ **Definition 17** ([16]). *Let $\mathcal{M} : \mathcal{X}^n \to Y$ be an algorithm that operates on a dataset. We say that $\mathcal{M}$ is $(\varepsilon, \delta, w_{\max})$-PSO secure if for every $w \le w_{\max}$, every distribution $\mathcal{D}$ over $\mathcal{X}$, and every adversary $\mathcal{A}$ it holds that*

$$\Pr_{\substack{S\leftarrow\mathcal{D}^n \\ y\leftarrow\mathcal{M}(S) \\ p\leftarrow\mathcal{A}(y)}}\left[\sum_{x\in S} p(x) = 1 \wedge \mathbf{E}_{\mathcal{D}}[p] \le w\right] \le e^{\varepsilon} \cdot \sup_{\substack{\text{predicate } p \\ \text{s.t. } \mathbf{E}_{\mathcal{D}}[p]\le w}} \left\{ n \cdot \mathbf{E}_{\mathcal{D}}[p] \cdot (1 - \mathbf{E}_{\mathcal{D}}[p])^{n-1} \right\} + \delta.$$

Using the paradigm of Narcissus resiliency, we present an alternative definition for predicate singling out, which is simpler in that it avoids reasoning directly about the expectations of the predicates.

▶ **Definition 18** (Narcissus singling out security). *Let $\mathcal{M} : \mathcal{X}^n \to Y$ be an algorithm. Let R be the relation that takes a dataset S and a predicate p, where $R(S,p) = 1$ if and only if p evaluates to 1 on* exactly *one point in S. We say that $\mathcal{M}$ is (ε, δ)-Narcissus-singling-out-secure if for every distribution $\mathcal{D}$ over $\mathcal{X}$ and for every attacker $\mathcal{A}$ it holds that*

$$\Pr_{\substack{S\leftarrow\mathcal{D}^n \\ y\leftarrow\mathcal{M}(S) \\ p\leftarrow\mathcal{A}(y)}}[R(S,p) = 1] \le e^{\varepsilon} \cdot \Pr_{\substack{S\leftarrow\mathcal{D}^n \\ T\leftarrow\mathcal{D}^n \\ y\leftarrow\mathcal{M}(S) \\ p\leftarrow\mathcal{A}(y)}}[R(T,p) = 1] + \delta.$$

Intuitively, Definitions 17 and 18 have similar interpretations: Finding a predicate that isolates a record in S after seeing the outcome of $\mathcal{M}$ is almost as hard as doing this without seeing the outcome of $\mathcal{M}$. However, the definitions are not equivalent. We leave open the question of understanding the relationships between these two definitions.

3 Narcissus-Resiliency Prevents Non-Trivial Extraction

Let $\mathcal{M}$ be a mechanism which is applied to a dataset S to obtain an outcome y. Let $\mathcal{B}$ be an "extraction attacker" that takes the outcome y and aims to compute a short program $\mathcal{A}$ that serves as an extraction evidence for S w.r.t. a relation R and a programming language $\mathcal{L}$. We show that if $\mathcal{M}$ is Narcissus-resilient (w.r.t. an appropriate reconstruction relation), then it prevents $\mathcal{B}$ from succeeding in its attack. Formally,

▶ **Definition 19.** *Fix a parameter $q \leq 1$ controlling the desired* quality *of extraction (as in Definition 13). We define the relation R_{ext} that takes a dataset S and a pair $(y, \mathcal{A})$ where y is an outcome of $\mathcal{M}$ and $\mathcal{A}$ is a program, and returns 1 if and only if the following two conditions hold:*

1. $R(S, \mathcal{A}(y)) = 1$, *and*
2. $\frac{|\mathcal{A}|}{K_{\mathcal{L}}(\{z\colon R(S,z)=1\})} \leq 1 - q$.

▶ **Lemma 20** (Narcissus-resiliency prevents non-trivial extraction)**.** *Let $\mathcal{M}$ be an $(\varepsilon, \delta, \mathcal{F})$-$R_{\text{ext}}$-Narcissus-resilient according to Definition 6, and consider an adversary $\mathcal{B}$ that takes the outcome of $\mathcal{M}$ and outputs an $\mathcal{L}$-program $\mathcal{A}$. Then for every $\mathcal{D} \in \mathcal{F}$ we have*

$$\Pr_{\substack{S\leftarrow\mathcal{D}\\ y\leftarrow\mathcal{M}(S)\\ \mathcal{A}\leftarrow\mathcal{B}(y)}}\left[\begin{array}{c}\mathcal{A} \text{ is an } (R,\mathcal{L})\text{-extraction}\\ \text{evidence of } S \text{ from } y\end{array}\right] \leq e^{\varepsilon}\cdot \Pr_{\substack{S\leftarrow\mathcal{D}\\ T\leftarrow\mathcal{D}\\ y\leftarrow\mathcal{M}(S)\\ \mathcal{A}\leftarrow\mathcal{B}(y)}}\left[\begin{array}{c}\mathcal{A} \text{ is an } (R,\mathcal{L})\text{-extraction}\\ \text{evidence of } T \text{ from } y\end{array}\right] + \delta$$

Proof. This follows directly from the assumption that $\mathcal{M}$ is R_{ext}-Narcissus-resilient. Formally, let $\hat{\mathcal{B}}$ denote the adversary that on input y returns $(y, \mathcal{B}(y))$. Then,

$$\begin{aligned}\Pr_{\substack{S\leftarrow\mathcal{D}\\ y\leftarrow\mathcal{M}(S)\\ \mathcal{A}\leftarrow\mathcal{B}(y)}}\left[\begin{array}{c}\mathcal{A} \text{ is an } (R,\mathcal{L})\text{-extraction}\\ \text{evidence of } S \text{ from } y\end{array}\right] &= \Pr_{\substack{S\leftarrow\mathcal{D}\\ y\leftarrow\mathcal{M}(S)\\ (y,\mathcal{A})\leftarrow\hat{\mathcal{B}}(y)}}\left[R_{\text{ext}}(S,(y,\mathcal{A}))=1\right]\\ &\leq e^{\varepsilon}\cdot \Pr_{\substack{S\leftarrow\mathcal{D}\\ T\leftarrow\mathcal{D}\\ y\leftarrow\mathcal{M}(S)\\ (y,\mathcal{A})\leftarrow\hat{\mathcal{B}}(y)}}\left[R_{\text{ext}}(T,(y,\mathcal{A}))=1\right]+\delta\\ &\leq e^{\varepsilon}\cdot \Pr_{\substack{S\leftarrow\mathcal{D}\\ T\leftarrow\mathcal{D}\\ y\leftarrow\mathcal{M}(S)\\ \mathcal{A}\leftarrow\mathcal{B}(y)}}\left[\begin{array}{c}\mathcal{A} \text{ is an } (R,\mathcal{L})\text{-extraction}\\ \text{evidence of } T \text{ from } y\end{array}\right]+\delta\end{aligned}$$ ◀

A possible downside of Lemma 20 is that it leverages a *different* reconstruction relation for Narcissus-resiliency compared to the extraction relation. This was needed in the proof because our definition of extraction considered two conditions: not only that the attacker needs to satisfy the relation R, but it needs to do it with a "short enough" program. To capture this in the above lemma, we incorporated this condition in the reconstruction relation used for Narcissus-resiliency. This can be avoided in cases where for some parameter k we have:

1. The adversary $\mathcal{B}$ always outputs a program of length at most k; and
2. The underlying data distribution $\mathcal{D}$ is such that with overwhelming probability over sampling $S \leftarrow \mathcal{D}$ we have that $\frac{k}{K_{\mathcal{L}}(\{z\colon R(S,z)=1\})} \leq 1 - q$.

That is, in cases when sampling a dataset $S \leftarrow \mathcal{D}$ then with overwhelming probability the Kolmogorov complexity $K_{\mathcal{L}}(\{z\colon R(S,z)=1\})$ is high. Indeed, if this is the case, then the second condition in Definition 19 holds with overwhelming probability, and hence can be ignored, which unifies the two relations R and R_{ext}.

3.1 Limitation of Lemma 20

It is important to note that Lemma 20 is limited to attackers $\mathcal{B}$ that only see the output y of the mechanism. But some of the real-world attacks also use the training data in order to construct an extraction evidence. E.g., in the Diffusion Model attack of [11] (see Section A.3), the attacker $\mathcal{B}$ first processes the input data in order to find captions of "interesting" images,

and then define $\mathcal{A}$ as the short program that simply queries the model on one of these captions. We note that Narcissus-resiliency does not prevent such attacks in general, because $\mathcal{B}$ uses the training data to generate $\mathcal{A}$. But Lemma 20 does imply that it is impossible to achieve non-trivial extraction of training examples, without knowing them in advanced.

References

1 Idan Attias, Gintare Karolina Dziugaite, Mahdi Haghifam, Roi Livni, and Daniel M. Roy. Information complexity of stochastic convex optimization: Applications to generalization and memorization, 2024. doi:10.48550/arXiv.2402.09327.

2 Achraf Azize and Debabrota Basu. How much does each datapoint leak your privacy? quantifying the per-datum membership leakage. *CoRR*, abs/2402.10065, 2024. doi:10.48550/arXiv.2402.10065.

3 Marshall Ball, Yanyi Liu, Noam Mazor, and Rafael Pass. Kolmogorov comes to cryptomania: On interactive kolmogorov complexity and key-agreement. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023*, pages 458–483, 2023. doi:10.1109/FOCS57990.2023.00034.

4 Borja Balle, Giovanni Cherubin, and Jamie Hayes. Reconstructing training data with informed adversaries. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1138–1156. IEEE, 2022. doi:10.1109/SP46214.2022.9833677.

5 Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In *Annual International Cryptology Conference (CRYPTO)*, pages 451–468, 2008. doi:10.1007/978-3-540-85174-5_25.

6 Gavin Brown, Mark Bun, Vitaly Feldman, Adam Smith, and Kunal Talwar. When is memorization of irrelevant training data necessary for high-accuracy learning? In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 123–132, 2021. doi:10.1145/3406325.3451131.

7 Gavin Brown, Mark Bun, and Adam Smith. Strong memory lower bounds for learning natural models. In *Proceedings of Thirty Fifth Conference on Learning Theory*, volume 178 of *Proceedings of Machine Learning Research*, pages 4989–5029, 2022. URL: https://proceedings.mlr.press/v178/brown22a.html.

8 Mark Bun, Yi-Hsiu Chen, and Salil P. Vadhan. Separating computational and statistical differential privacy in the client-server model. In *Theory of Cryptography - 14th International Conference, TCC 2016-B*, volume 9985, pages 607–634, 2016. doi:10.1007/978-3-662-53641-4_23.

9 Mark Bun and Mark Zhandry. Order-revealing encryption and the hardness of private learning. In *TCC 2016-A*, volume 9562, pages 176–206, 2016. doi:10.1007/978-3-662-49096-9_8.

10 Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1897–1914, 2022. doi:10.1109/SP46214.2022.9833649.

11 Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *Proceedings of the 32nd USENIX Conference on Security Symposium*, SEC '23, 2023.

12 Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: evaluating and testing unintended memorization in neural networks. In *Proceedings of the 28th USENIX Conference on Security Symposium*, SEC'19, pages 267–284, USA, 2019. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/carlini.

13 Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium, USENIX Security 2019*, pages 267–284, 2019. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/carlini.

14 Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.

15 Gregory J. Chaitin. On the simplicity and speed of programs for computing infinite sets of natural numbers. *J. ACM*, 16(3):407–422, 1969. doi:10.1145/321526.321530.

16 Aloni Cohen and Kobbi Nissim. Towards formalizing the gdpr's notion of singling out. *Proc. Natl. Acad. Sci. USA*, 117(15):8344–8352, 2020. doi:10.1073/PNAS.1914598117.

17 Edith Cohen, Haim Kaplan, Yishay Mansour, Shay Moran, Kobbi Nissim, Uri Stemmer, and Eliad Tsfadia. Data reconstruction: When you see it and when you don't, 2024. doi:10.48550/arXiv.2405.15753.

18 Rachel Cummings, Shlomi Hod, Jayshree Sarathy, and Marika Swanberg. Attaxonomy: Unpacking differential privacy guarantees against practical adversaries, 2024. doi:10.48550/arXiv.2405.01716.

19 Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210. ACM, 2003. doi:10.1145/773153.773173.

20 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, volume 3876, pages 265–284, 2006. doi:10.1007/11681878_14.

21 Cynthia Dwork, Frank McSherry, and Kunal Talwar. The price of privacy and the limits of lp decoding. In *STOC*, pages 85–94. ACM, 2007. doi:10.1145/1250790.1250804.

22 Cynthia Dwork, Adam D. Smith, Thomas Steinke, Jonathan R. Ullman, and Salil P. Vadhan. Robust traceability from trace amounts. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*, pages 650–669, 2015. doi:10.1109/FOCS.2015.46.

23 Cynthia Dwork and Sergey Yekhanin. New efficient attacks on statistical disclosure control mechanisms. In *CRYPTO*, pages 469–480. Springer, 2008. doi:10.1007/978-3-540-85174-5_26.

24 Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 954–959, 2020. doi:10.1145/3357713.3384290.

25 Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: discovering the long tail via influence estimation. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, 2020.

26 Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1322–1333, 2015. doi:10.1145/2810103.2813677.

27 B. Ghazi, R. Ilango, P. Kamath, R. Kumar, and P. Manurangsi. Towards separating computational and statistical differential privacy. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 580–599, 2023.

28 Peter D. Grünwald and Paul M. B. Vitányi. Kolmogorov complexity and information theory. with an interpretation in terms of questions and answers. *Journal of Logic, Language and Information*, 12(4):497–529, 2003. doi:10.1023/A:1025011119492.

29 Chuan Guo, Brian Karrer, Kamalika Chaudhuri, and Laurens van der Maaten. Bounding training data reconstruction in private (deep) learning. In *International Conference on Machine Learning, ICML 2022*, volume 162, pages 8056–8071, 2022. URL: https://proceedings.mlr.press/v162/guo22c.html.

30 Niv Haim, Gal Vardi, Gilad Yehudai, Ohad Shamir, and Michal Irani. Reconstructing training data from trained neural networks. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems, NeurIPS 2022*, 2022.

31 Iftach Haitner, Noam Mazor, Jad Silbak, and Eliad Tsfadia. On the complexity of two-party differential privacy. In *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1392–1405. ACM, 2022. doi:10.1145/3519935.3519982.

32 Jamie Hayes, Borja Balle, and Saeed Mahloujifar. Bounding training data reconstruction in DP-SGD. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

33 Zecheng He, Tianwei Zhang, and Ruby B. Lee. Model inversion attacks against collaborative inference. In *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19, pages 148–162, 2019. doi:10.1145/3359789.3359824.

34 Matthew Jagielski, Om Thakkar, Florian Tramer, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Guha Thakurta, Nicolas Papernot, and Chiyuan Zhang. Measuring forgetting of memorized training examples. In *The Eleventh International Conference on Learning Representations*, 2023.

35 Georgios Kaissis, Jamie Hayes, Alexander Ziller, and Daniel Rueckert. Bounding data reconstruction attacks with the hypothesis testing interpretation of differential privacy, 2023. doi:10.48550/arXiv.2307.03928.

36 K.-I. Ko. On the notion of infinite pseudorandom sequences. *Theor. Comput. Sci.*, 48(1):9–33, 1986. doi:10.1016/0304-3975(86)90081-2.

37 A. N. Kolmogorov. Three approaches to the quantitative definition of information. *International Journal of Computer Mathematics*, 2(1-4):157–168, 1968.

38 L Levin. Universal search problems (russian), translated to english by trakhtenbrot (1984). *Problems of Information Transmission*, 9(3):265–266, 1973.

39 Yanyi Liu and Rafael Pass. On one-way functions and kolmogorov complexity. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 1243–1254, 2020. doi:10.1109/FOCS46700.2020.00118.

40 Yanyi Liu and Rafael Pass. On the possibility of basing cryptography on exp$\neq$ BPP. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021*, volume 12825, pages 11–40, 2021. doi:10.1007/978-3-030-84242-0_2.

41 Yanyi Liu and Rafael Pass. Characterizing derandomization through hardness of levin-kolmogorov complexity. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022*, volume 234, pages 35:1–35:17, 2022. doi:10.4230/LIPICS.CCC.2022.35.

42 Yanyi Liu and Rafael Pass. On one-way functions from np-complete problems. In *37th Computational Complexity Conference, CCC 2022*, volume 234, pages 36:1–36:24, 2022. doi:10.4230/LIPICS.CCC.2022.36.

43 Roi Livni. Information theoretic lower bounds for information theoretic upper bounds. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023*, 2023.

44 Yunhui Long, Lei Wang, Diyue Bu, Vincent Bindschaedler, Xiaofeng Wang, Haixu Tang, Carl A. Gunter, and Kai Chen. A pragmatic approach to membership inferences on machine learning models. In *2020 IEEE European Symposium on Security and Privacy*, pages 521–534, 2020. doi:10.1109/EUROSP48549.2020.00040.

45 Luc Longpré and Sarah Mocas. Symmetry of information and one-way functions. *Information Processing Letters*, 46(2):95–100, 1993. doi:10.1016/0020-0190(93)90204-M.

46 Jean loup Gailly and Mark Adle. zlib compression library, 2004.

47 Andrew McGregor, Ilya Mironov, Toniann Pitassi, Omer Reingold, Kunal Talwar, and Salil Vadhan. The limits of two-party differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 81–90, 2010. doi:10.1109/FOCS.2010.14.

48 Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. Computational differential privacy. In *Annual International Cryptology Conference (CRYPTO)*, pages 126–142, 2009. doi:10.1007/978-3-642-03356-8_8.

49 Kobbi Nissim, Aaron Bembenek, Alexandra Wood, Mark Bun, Marco Gaboardi, Urs Gasser, David R. O'Brien, Thomas Steinke, and Salil Vadhan. Bridging the gap between computer science and legal approaches to privacy. *Harvard Journal of Law & Technology*, 31(2):687–780, 2018.

50 Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Hervé Jégou. White-box vs black-box: Bayes optimal strategies for membership inference. In *International Conference on Machine Learning*, 2019.

51 Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems, NeurIPS 2022*, 2022.

52 Sriram Sankararaman, Guillaume Obozinski, Michael I Jordan, and Eran Halperin. Genomic privacy and limits of individual detection in a pool. *Nature genetics*, 41(9):965–967, 2009.

53 Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017. doi:10.1109/SP.2017.41.

54 Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy, SP 2017*, pages 3–18, 2017. doi:10.1109/SP.2017.41.

55 Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, pages 330–335, 1983. doi:10.1145/800061.808762.

56 R.J. Solomonoff. A formal theory of inductive inference. *Information and Control*, 7(1):1–22, 1967.

57 B.A. Trakhtenbrot. A survey of russian approaches to perebor (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984. doi:10.1109/MAHC.1984.10036.

58 Lauren Watson, Chuan Guo, Graham Cormode, and Alexandre Sablayrolles. On the importance of difficulty calibration in membership inference attacks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022. URL: https://openreview.net/forum?id=3eIrli0TwQ.

59 Ziqi Yang, Jiyi Zhang, Ee-Chien Chang, and Zhenkai Liang. Neural network inversion in adversarial setting via background knowledge alignment. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, pages 225–240. Association for Computing Machinery, 2019. doi:10.1145/3319535.3354261.

60 Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, pages 268–282. IEEE, 2018. doi:10.1109/CSF.2018.00027.

61 H. Yin, P. Molchanov, J. M. Alvarez, Z. Li, A. Mallya, D. Hoiem, N. K. Jha, and J. Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8712–8721. IEEE Computer Society, 2020.

62 A Zvonkin and L Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, 6:83–124, 1970.

A Recognizing Extraction

Following the discussion in Section 1.2, we use the terminology of "extraction" rather than "reconstruction" in the setting where the dataset S and the model y are fixed. We formally define how to recognize a valid extraction attack on a given (fixed) model y (as described in Section 1.2), use the definition to explain the validity of some real-world attacks, and prove that in general the problem of verifying a valid attack is computationally hard assuming that cryptography exists.

A.1 Defining Extraction via Kolmogorov Complexity

In this paper, we think of extraction as a randomized process that can fail with some probability, so we use the following variant of Definition 12.

▶ **Definition 21** ($K_{\mathcal{L}}$-Complexity, a probabilistic version of Definition 12)**.** *Let $\mathcal{L}$ be a programming language. The $K_{\mathcal{L}}$-complexity of a string x, denoted by $K_{\mathcal{L}}(x)$, is the length of the shortest $\mathcal{L}$-program that with probability $2/3$ outputs the string x and halts.*

In theoretical results, this complexity is usually defined w.r.t. a fixed *Universal Turing Machine* $\mathcal{U}$ that gets a description $\mathcal{M}$ of a Turing machine and outputs $\mathcal{U}(\mathcal{M})$. Then, $K(x) = K_{\mathcal{U}}(x)$ is defined by the shortest $\mathcal{M}$ such that $\Pr[\mathcal{U}(\mathcal{M}) = x] \geq 2/3$. This is good enough for *asymptotic* analysis since for any programming language $\mathcal{L}$ and every x it holds that $K(x) \leq K_{\mathcal{L}}(x) + O(1)$, where $O(1)$ denotes a constant that is independent of $|x|$.[6] However, when we would like to deal with concrete quantities (where constants matter), we must explicitly specify a concrete programming language $\mathcal{L}$, and therefore we do not omit it from the notation.

After fixing the programming language $\mathcal{L}$, the quantity $K_{\mathcal{L}}(x)$ can be thought of as the analogous to the "entropy" of the string x. As with entropy, we can also consider a conditional version of the K-complexity (the analog of conditional entropy) as well as "mutual K-information" (the analog of mutual information) [62, 38, 57, 45, 28, 41, 3]:

▶ **Definition 22** (Conditional $K_{\mathcal{L}}$-Complexity, and mutual $K_{\mathcal{L}}$-information)**.** *Given a programming language $\mathcal{L}$ and two strings x, y, we define the* conditional $K_{\mathcal{L}}$-complexity of x given y*, denoted by $K_{\mathcal{L}}(x \mid y)$, as the length of the shortest program in $\mathcal{L}$ that given y as input, with probability $2/3$ outputs x and halts. We define the* mutual $K_{\mathcal{L}}$-information *of x, y as $KI_{\mathcal{L}}(x; y) = K_{\mathcal{L}}(x) - K_{\mathcal{L}}(x \mid y)$.*

Intuitively, $KI_{\mathcal{L}}(x; y)$ is high if it is possible to extract x from y using a short $\mathcal{L}$-program, but it is not possible to do so without y. In our context of extracting an input example x from a model y, we should also allow to output elements z that are "similar" to x (which are considered as a valid extraction of x under some metric), or perhaps the goal is to output a list of elements that some of them are close to training examples. Therefore, we would be interested in an extension of the K-complexity into a set of strings which allows much more flexibility.

▶ **Definition 23** (Extension of Definitions 21 and 22 for sets)**.** *Let $\mathcal{L}$ be a programming language. The $K_{\mathcal{L}}$-complexity of a set of strings X, denoted by $K_{\mathcal{L}}(X)$, is the length of the shortest $\mathcal{L}$-program that with probability $2/3$, outputs a string $x \in X$ and halts. Similarly to Definition 22, we define $K_{\mathcal{L}}(X \mid y)$ as the length of the shortest $\mathcal{L}$-program that given an input y, outputs w.p. $2/3$ a string $x \in X$ and halts, and we define $KI_{\mathcal{L}}(X; y) = K_{\mathcal{L}}(X) - K_{\mathcal{L}}(X \mid y)$.*

Note that by definition, $KI_{\mathcal{L}}(X; y) = \min_{x \in X} K_{\mathcal{L}}(x) - \min_{x \in X} K_{\mathcal{L}}(x \mid y)$, and these minimums are not necessarily realized by the same x.

A few examples are in order. In all the following examples, we use $\mathcal{L}$ as a fixed Universal Turing Machine language, and for $x = (x^1, \ldots, x^d), z = (z^1, \ldots, z^d) \in \{0,1\}^d$ we denote by $\mathsf{NHamDist}(x, z) = \frac{1}{d} \cdot \left|\{i \in [n] \colon x^i \neq z^i\}\right|$ the *Normalized Hamming Distance* between x and z.

---

[6] Let $\mathcal{C}$ be a $\mathcal{U}$-program (i.e., a Turing machine description) that serves as a compiler from a language $\mathcal{L}$ to $\mathcal{U}$, i.e., give a $\mathcal{L}$-program $\mathcal{M}$, $C(\mathcal{M})$ outputs a $\mathcal{U}$-program $\mathcal{M}'$ that acts as $\mathcal{M}$. So for every x, if it can be described using an $\mathcal{L}$-program $\mathcal{M}$, then it can be described using a $\mathcal{U}$-program $\mathcal{C}(\mathcal{M})$ of length $|\mathcal{M}| + |\mathcal{C}| = |\mathcal{M}| + O(1)$ (note that $|\mathcal{C}|$ is independent of $|x|$).

▶ **Example 24.** For $x \in \{0,1\}^d$ define $R_x = \{(z_1, z_2) \in \{0,1\}^{2d} : \min_{i\in\{1,2\}} \mathsf{NHamDist}(x, z_i) \leq 1/2\}$, (i.e., in this example, R_x defines the task of outputting two strings that at least one of them agrees with x on at least half of the bits). Note that $K_{\mathcal{L}}(R_x) = O(\log d)$ for any x, because the program that outputs a uniformly random $(z_1, z_2) \leftarrow \{0,1\}^{2d}$ – e.g., a for loop executing for $2d$ iterations and outputing a random bit in each – satisfies $(z_1, z_2) \in R_x$ with probability $3/4$.

▶ **Example 25.** For $x \in \{0,1\}^d$ define R_x as the set of all the tuples of the form $z = (z_1, \ldots, z_m)$, for $m \leq \text{poly}(d)$ (for some fixed polynomial), such that $\min_{i\in[m]} \mathsf{NHamDist}(x, z_i) \leq 1/4$ (i.e., at least one of the z_i's agrees with $3/4$ of the bits of x). For a uniformly random $x \leftarrow \{0,1\}^d$, it holds with high probability that $K_{\mathcal{L}}(R_x) \geq (1 - o(1))d/2$ (i.e., for most strings x, any algorithm that aims to output a string that agrees with $3/4$ of the bits of x, essentially must memorize $\approx d/2$ bits of information about x). Indeed, given $y = (x^1, \ldots, x^{d/2})$ (the first half of the bits of x) as input, we have $K(R_x \mid y) = O(\log d)$ because the algorithm that chooses $m = 2$ uniformly random strings $(z_1, z_2) \leftarrow \{0,1\}^{d/2}$ and outputs $z = (y \circ z_1,\ y \circ z_2)$ satisfies $z \in R_x$ with probability $3/4$.

▶ **Example 26.** Let $S = (x_1, \ldots, x_n) \in (\{0,1\}^d)^n$, and define R_S as the set of all the tuples $z = (z_1, \ldots, z_m)$ such that there exist $i \in [n]$ and $j \in [m]$ with $\mathsf{NHamDist}(x_i, z_j) \leq 1/4$. Assume $n, m \leq \text{poly}(d)$ (for some fixed polynomial). Then as in Example 25, it can be shown that when S consists of uniformly random strings, then $K_{\mathcal{L}}(R_S) \geq (1 - o(1))d/2$ with high probability, meaning that solving this problem requires to memorize at least $\approx d/2$ bit of information from one of the strings.

We are now ready to reformulate the extraction definitions of [14, 11].

▶ **Definition 27** (Extractable Information)**.** *Let S be a set of elements over a domain $\mathcal{X}$, let $R : \mathcal{X}^* \times \{0,1\}^* \to \{0,1\}$ be an extraction relation (i.e., $R(S, z) = 1$ means that z is considered a valid extraction of elements in S), let $\mathcal{L}$ be a programming language, and let $R_S = \{z : R(S, z) = 1\}$. We say that a model y contains $(R, \mathcal{L}, \tau)$-extractable information about S if*

$$KI_{\mathcal{L}}(R_S; y) \geq \tau.$$

In order to prove that S is extractable from a model y, it suffices to present an extractor:

▶ **Definition 28** (Extractor, redefinition of Definition 13)**.** *Let $R : \mathcal{X}^* \times \{0,1\}^* \to \{0,1\}$ be an extraction relation and denote $R_S = \{z : R(S, z) = 1\}$. Let $q \in [0, 1)$ be a quality parameter. We say that an $\mathcal{L}$-program $\mathcal{A}$ is an $(\mathcal{L}, R, q)$-extractor of $S \in \mathcal{X}^*$ from y if the following holds:*

1. $\Pr[\mathcal{A}(y) \in R_S] \geq 2/3$*, and*
2. $K_{\mathcal{L}}(R_S) \geq \frac{|\mathcal{A}|}{1-q}$.

Note that when q is closer to 1 is means a more significant extraction.

▷ Claim 29. If there exists an $(\mathcal{L}, R, q)$-extractor $\mathcal{A}$ of S from y, then y contains $(\mathcal{L}, R, \tau = \frac{q}{1-q} \cdot |\mathcal{A}|)$-extractable information about S (Definition 27).

Proof. Compute

$$KI_{\mathcal{L}}(R_S; y) = K_{\mathcal{L}}(R_S) - K_{\mathcal{L}}(R_S \mid y) \geq \frac{|\mathcal{A}|}{1-q} - |\mathcal{A}| = \frac{q}{1-q} \cdot |\mathcal{A}|. \qquad \triangleleft$$

A.2 Capturing Interesting Extractions using the Predicate

In the previous examples (e.g., Example 26), we considered simple relations R that only check for a small distance from an example point. But in the real-world, when thinking about S as the training dataset and on y as the trained model (over S), it is very likely that S will contain duplicated data, and in such cases, it might be considered ok to memorize such data. As a concrete example, if we think about a chat-bot model, then we can query it with "Print the Bible" and it will likely do that. I.e., even though it is indeed a valid extraction (as the K-complexity of the Bible is large), this is not an interesting one since the Bible content is probably duplicated in many training examples. In order to define more interesting relations R, it is reasonable to consider extraction of only *k-Eidetic Memorizated* strings x ([14]) for some small threshold k, which means that such x's only appear at most k times in S.

A.3 Real-World Examples

In order to illustrate the generality of our definitions, we next pick three different types of real-world attacks, and explain what is happening in each of them in terms of Definition 28.

Extraction from Large Language Models [14]

[14] present extraction attacks on GPT-2 by essentially generating many samples from GPT-2 when the model is conditioned on (potentially empty) prefixes. Then they sort each generation according to some metric to remove duplications, and this gives a set of potentially memorized texts from training examples. In one of their methods, they choose the top-m interesting ones by assigning a score to each such text which combines the compressibility of the text using some compression algorithm like zlib [46], and the likelihood of it, e.g., using a "perplexity" measure [13] (higher compressibility and likelihood increase the score). They show that noticeable fraction of such strings are "Eidetic Memorizated", meaning that they only appear a small number of times in the training data (which makes them more interesting).

In terms of our extraction methodology, the training dataset $S = (x_1, \ldots, x_n)$ consists of webpages, and an *interesting extractable text* is a string s that: (1) has high entropy (i.e., high K-complexity, which can be estimated using heuristics like zlib compression) and (2) appears only small number of times in the training examples (here, "appear" in webpage x_i means $s \subseteq x_i$). The attack outputs a list of strings $z = (z_1, \ldots, z_m)$, and succeed when there exists a sub-list $(z_{j_1}, \ldots, z_{j_\ell})$ such that each string z_{j_i} is interesting and extractable, and the K-complexity of the entire list is higher than some threshold (which, again, can be estimated using heuristics like zlib compression).

When the K-complexity of the resulting sub-list is much higher than the length of the attacking code that extracted this information from the GPT-2 model, the attack is considered to have high quality.

Extraction from Diffusion Models [11]

[11] present several extraction attacks on Diffusion models, where all the attacks assume that the training data is given in advance. As one concrete example, they managed to extract training images from Imagen ([51]), a 2 billion parameter text-to-image diffusion model, by first searching for the captions of "outliers" in the training data (that is, training images that are less similar to other training images according to some metric), and then showed that when querying the model on such captions, some of them result with a good approximation of unique training images.

In terms of our extraction methodology, the end program that queries the model on a specific caption that leads to extraction of a unique training image is extremely short compared to the K-complexity of the image that was extracted. So this should be considered a very high-quality extraction to a predicate of the following type (for some threshold parameters $T > t$):

$$R(S = (x_1, \ldots, x_n), z) = 1 \iff \exists i \in [n] \text{ s.t. } (\ell_2(x_i, z) < t \text{ and } \forall j \in [n] \setminus \{i\} : \ell_2(x_i, x_j) > T).$$

Extraction from Binary Classifiers trained by Large Neural Networks [30]

[30] present a general attack that works on large class of binary classifiers trained by Neural networks with many parameters p. Roughly, their attack works as follows: Given the parameters of the model $y \in \mathbb{R}^p$, they define an optimization problem (that depends on y) on variables $z_1, \ldots, z_m \in \mathbb{R}^d$ and $\lambda_1, \ldots, \lambda_m \in \mathbb{R}$ and optimize it, and show that in the regime $p > n \cdot d$, the solution $z_1, \ldots, z_m$ is likely to contain a good approximation of many data points.

As one example of their experimental evaluation, they trained the model on the training images of CIFAR10 on vehicles vs. animals classification, and showed that their optimization program, which is defined based on the model $y \in \mathbb{R}^p$ given as input, managed to extract a good approximation of many training images (the top 45 extractions are presented in their paper).

In terms of our extraction methodology, given the training data $S = (x_1, \ldots, x_n)$, we can define the extraction relation

$$R(S = (x_1, \ldots, x_n), z = (z_1, \ldots, z_m)) = 1 \iff |\{i \in [n] \colon \exists j \in [m] \text{ s.t. } z_j \text{ is "close" to } x_i\}| \geq t,$$

for some parameter t (say, $t = 45$), where "close" is according to some metric. Under the reasonable assumption that $K(R_S)$ is much higher than the length of their attack (which is given the model y as input), this is considered a high quality extraction.

A.4 Inherent Limitations of Definition 28

The main limitation of our Definition 28 is that Item 2 (lower-bounding the Kolmogorov Complexity) is not verifiable, as computing the K-complexity is an intractable problem. While in practice the K-complexity can be estimated using Heuristics like zlib compression [46], any such Heuristic can fail to determine some highly compressible patterns, and as we demonstrate next in Section A.4.1, this problem is indeed inherent assuming that basic cryptography exists.

A well studied tractable variant of the K-complexity is the $t(\cdot)$-time bounded K-complexity, denoted by K^t-complexity, where $K^t(x)$ is the length of the shortest program that outputs x within $t(|x|)$ steps ([37, 55, 57, 36]). However, there is no efficient algorithm for computing the K^t-complexity.[7] Very recently, in a sequence of breakthrough results in cryptography and complexity theory [39, 40, 42, 41], it was shown that (a close variant of) this problem is hard if and only if one-way functions, the most basic cryptographic primitives, exist. Since a real-world verifier is a polynomial-time algorithm, this means that switching from

[7] As surveyed by [57], the problem of efficiently determining the K^t-complexity for $t(n) = \text{poly}(n)$ predates the theory of NP-completeness and was studied in the Soviet Union since the 60s as a candidate for a problem that requires "brute-force search" (see Task 5 on page 392 in [57]).

$K(\cdot)$ to the $K^{\mathrm{poly}(n)}(\cdot)$ variant will essentially not make any difference. Actually, the hardness of verification is inherent, and in the next two sections we show, under common cryptographic assumptions, how to construct triplets $(\mathcal{A}, x, y)$, with efficient $\mathcal{A}$, such that it is computationally hard to verify whether $\mathcal{A}$ extracts information about x from y.

A.4.1 Hardness of Verification

Our harness example is based on the existence of Pseudorandom Generator (PRG).

▶ **Definition 30** (Pseudorandom Generator (PRG)). *A PRG is a* PPT *algorithm G that maps d bits to $\ell(d)$ bits (for $\ell(d) > d$) such that for every* PPT *algorithm V and every $d \in \mathbb{N}$ it holds that*

$$\Pr[V(G(\mathcal{U}_d)) = 1] \leq \Pr\left[V(\mathcal{U}_{d+\ell(d)}) = 1\right] + \mathrm{neg}(d),$$

where $\mathcal{U}_m$ denotes the uniform distribution over $\{0,1\}^m$.

▶ **Lemma 31.** *Assume there exists a PRG G that maps d bits to d^4 bits. Let $\mathcal{L}$ be a Universal Turing Machine programming language with oracle access to G. Then there exists a* PPT *algorithm* Attacker *such that on input 1^d and $b \in \{0,1\}$, outputs a deterministic program $\mathcal{A}$ and two strings $x, y \in \{0,1\}^{d^4}$ such that the following holds w.r.t. a random execution $(\mathcal{A}, x, y) \leftarrow \mathsf{Attacker}(1^d, b)$:*

1. *$|\mathcal{A}| = d^2 + O(1)$ and $\mathcal{A}(y) = x$ (which is an evidence for $K_{\mathcal{L}}(x \mid y) \leq d^2 + O(1)$).*
2. *If $b = 0$ then $K_{\mathcal{L}}(x) \leq d + O(1)$ (and in particular, there is a deterministic $\mathrm{poly}(d)$-time $(d + O(1))$-size algorithm that outputs x).*
3. *If $b = 1$ then x is chosen uniformly over $\{0,1\}^{d^4}$ (which yields $K_{\mathcal{L}}(x) \geq (1 - o(1))d^4$ w.h.p.),*
4. *For every* PPT *$\mathcal{V}$ it holds that*

$$\Pr_{b \leftarrow \{0,1\},\, (\mathcal{A},x,y) \leftarrow \mathsf{Attacker}(1^d,b)}[\mathcal{V}(\mathcal{A}, x, y) = b] \leq 1/2 + \mathrm{neg}(d).$$

Note that when $b = 1$, $\mathcal{A}$ extracts most of the information about x from y (quality $1 - \frac{1+o(1)}{d^2}$), while the case $b = 0$ is clearly not an extraction as $K_{\mathcal{L}}(x) \ll |\mathcal{A}|$. Yet, by Item 4, any PPT verifier $\mathcal{V}$ who only sees $\mathcal{A}, x, y$ cannot distinguish between the two cases.

Proof. The proof holds by considering the following Attacker algorithm.

Algorithm 1 Attacker.

- Input: 1^d and $b \in \{0,1\}$.
- Oracle: PRG G: $\{0,1\}^d \to \{0,1\}^{d^4}$.
- Operation:
  1. If $b = 0$: Sample $s \leftarrow \{0,1\}^d$ and let $x = (x_1, \ldots, x_{d^4}) = G(s)$.
  2. Else (i.e., $b = 1$): Sample $x = (x_1, \ldots, x_{d^4}) \leftarrow \{0,1\}^{d^4}$.
  3. Let $y = (x_1, \ldots, x_{d^4 - d^2})$.
  4. Let $\mathcal{A}$ be the $(d^2 + O(1))$-size algorithm that has $x' = (x_{d^4-d^2+1}, \ldots, x_{d^4})$ hard-coded, and given y as input, outputs $y \circ x'$ and halts.
  5. Output $(\mathcal{A}, x, y)$.

Note that Items 1–3 of the lemma holds by construction. Furthermore, Item 4 immediately holds since G is a PRG. ◀

In the above example, we defined a hardness problem w.r.t. the predicate $R(x, z) = 1 \iff z = x$. But it can be easily be relaxed to a more realistic predicate of the form

$$R(S = (x_1, \ldots, x_n), z = (z_1, \ldots, z_m)) = 1 \iff \exists i \in [n], j \in [m] \text{ s.t. } \{\mathsf{NHamDist}(x_i, z_j)\} \leq 1/2 - \alpha,$$

for any fixing of a constant $\alpha > 0$ and $n, m \leq \text{poly}(d)$ (i.e., when the task is to output a list of strings that at least one of them has non-trivial agreement with one of the elements of S).[8]

[8] The idea is to replace Step 3 of Attacker with $y = (x_1, \ldots, x_{\alpha d^4 - d^2 \log d})$, where x is one of the elements in $\mathcal{S}$, and in Step 4 to define $\mathcal{A}$ as the $O(d^2 \log d)$-size program that has $x' = (x_{\alpha d^4 - d^2 \log d}, \ldots, x_{\alpha d^4 + d^2 \log d})$ hard-coded, and given y as input, samples $x'' \leftarrow \{0,1\}^{(1-\alpha)d^4 - d^2 \log d}$ and outputs $z = y \circ x' \circ x''$. By concentration bounds of the binomial distribution, it holds that $\Pr[\mathsf{NHamDist}(x, z) \leq 1/2 - \alpha] \geq 1 - \text{neg}(d)$. But without memorizing x' or at least $\Omega(d^2 \log d)$ bits of information about one $x \in \mathcal{S}$, the probability that a random guess of the bits after y would lead to $1/2 - \alpha$ Normalized Hamming distance from an element $x \in \mathcal{S}$ is $\text{neg}(d)$.